Run-Time Support for Fine-Grain, Power-Constrained Exascale Computing[*]

Barry Rountree[†], Martin Schulz[†], David K. Lowenthal[‡]

Abstract

Power is a critical issue in achieving exascale performance. This will motivate the discipline of *power-constrained*, high-performance computing, where the goal is to optimize performance under a hard power limit. Optimization will require fine-grain control of power, coordinated among run-time systems, the operating system and the system hardware.

Introduction

One of the critical problems—if not *the* critical problem—in reaching the exascale computing goal by the end of the decade is the power problem. Exascale systems have a target power constraint of 20 megawatts even though today's petascale systems—which have performance at least two orders of magnitude below prospective exascale systems—consume more than 5 megawatts. While hardware improvements will bridge some of the gap, they will not nearly bridge all of it; the task of achieving exascale performance within a 20 megawatt power budget is quite daunting. It will be impossible for evolutionary approaches to achieve sustained exascale performance within realistic power bounds except perhaps on the most trivial of applications. This goal instead requires significant, disruptive changes to all aspects of high-performance computing, including the machine room, hardware, software, and programming—a significant undertaking. These changes include modifications to make operating systems and run-time systems power-conscious.

Fortunately, architectures are moving toward much more fine-grain control of power. For example, recent machines such as Sandy Bridge provide power capping; that is, it can allocate a prescribed amount of power to different subsystems. Currently, power can be specified for the (entire) core and the memory.

Research Approach

We envision a future where the power supplied to most chip resources is under software control. Therefore, work is necessary to provide support for *fine-grain, power-constrained, high-performance computing*. Assume that the power limitation for an application is pre-specified; how system administrators determine that limit is orthogonal to our work. Given the limit, we assume that the program can be executed in any *configuration*, which is defined as a tuple (number of nodes, number of cores per node, and how much power is allocated to each component of the chip). The constraint is that the total power implied by a configuration must be below the power limit. One can imagine that at a coarse-grain level, scalable applications will likely want more nodes at less power per node, and less scalable applications

[*]This work was partially performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-ABS-564233).

[†]Lawrence Livermore National Laboratory

[‡]Dept. of Computer Science, The University of Arizona

will likely want fewer nodes at more power per node. In addition, at a fine-grain level, the allocation of power to different chip resources will depend on application characteristics.

The first step is to analyze and provide support for fine-grain power control. Currently it is not possible to control individual resources within a core; and, it is also not known how much power is consumed by different resources. Software specialists should work with low-level hardware teams to study a large range of applications and their fine-grain power consumption. Studies to date in this area are too coarse-grain (covering just the core, memory, and disk, generally). This will hopefully drive hardware design for fine-grain control of resources, both in which resources need their power to be controlled as well as which resources should be the focus of low-power hardware research.

The second step is to develop comprehensive system software to find near-optimal configurations. This software should include both modeling and run-time frameworks that target fine-grain, power-controllable architectures. Modeling will be used to determine effective configurations, and then these will be effected by the run-time system. The run-time system will also be tasked with collecting the key information that is fed back into the modeling component. We also intend to investigate the use of programmer annotations to improve performance where our system software simply needs a small amount of extra information.

Related Work

The state of the art in transparent power/energy optimization involves primarily trading execution time for lower power/energy and doing so in a coarse-grain manner [1, 8]. Additionally, several have developed analytic models to predict or to understand energy consumption in the context of scalability [15, 5, 11]. Also, recent work has explored reducing the amount of concurrency in programs, with one of the benefits of such reduction being lower energy [4, 3, 9, 10]. Finally, there is significant work in annotations, such as those in the OpenMP specification [12], those for performance [13, 16], and those for data distribution [6, 7, 14, 17, 2]).

Assessment

In this section we assess our proposed research.

- *Challenges addressed.* The power problem is believed by many to be the critical problem to achieving exascale.
- *Maturity.* There has been a long line of work in using DVFS to save energy in HPC programs. That work has been reasonably successful. However, it is coarse-grain; CPU frequency is decreased and increased. This work will focus on optimizing performance rather than saving energy, and do so in a fine-grain manner.
- *Uniqueness.* There are many areas of computer science as well as society in which power is a critical resource. However, exascale computing is unique in that the concern is not trying to save power, but rather using all the available power (but no more) to optimize performance. Other research programs have not been generally interested in this problem.
- *Novelty.* As stated above, while there is significant work in saving energy, we do not know of work to optimize performance, in a fine-grain manner, under a power bound.
- *Applicability.* Building support for fine-grain power control will help achieve the exascale goal. However, it is also useful for allowing smaller organizations to achieve petascale. In addition, it could be useful for embedded systems.
- *Effort.* Investigating this approach is a multi-year effort. It should bring together experts in low-power hardware design and run-time system construction.

References

[1] K. W. Cameron, X. Feng, and R. Ge. Performance-constrained distributed DVS scheduling for scientific applications on power-aware clusters. In *Supercomputing*, Seattle, Washington, Nov. 2005.

[2] B. M. Chapman, H. Herbeck, and H. P. Zima. Automatic support for data distribution. In *Proceedings of the Sixth Distributed Memory Computing Conference*, Portland, Oregon, May 1991.

[3] M. Curtis-Maury, J. Dzierwa, C. D. Antonopoulos, and D. S. Nikolopoulos. Online power-performance adaptation of multithreaded programs using hardware event-based prediction. In *International Conference on Supercomputing*, 2006.

[4] Y. Ding, M. Kandemir, P. Raghavan, and M. Irwin. A helper thread based EDP reduction scheme for adapting application execution in CMPs. In *International Parallel and Distributed Processing Symposium (IPDPS)*, 2008.

[5] R. Ge, X. Feng, W. Feng, and K. W. Cameron. CPU Miser: A performance-directed, run-time system for power-aware clusters. In *Proceedings of the 2007 International Conference on Parallel Processing*, Xi'An, China, 2007.

[6] High Performance Fortran Forum. High Performance Fortran language specification, version 1.0. Technical Report CRPC-TR92225, Rice University, Houston, Tex., 1993.

[7] S. Hiranandani, K. Kennedy, and C.-W. Tseng. Preliminary experiences with the Fortran D compiler. In *Supercomputing*, Nov. 1993.

[8] C.-H. Hsu and W.-C. Feng. A power-aware run-time system for high-performance computing. In *Supercomputing*, Nov. 2005.

[9] D. Li, B. R. de Supinski, M. Schulz, K. W. Cameron, and D. S. Nikolopoulos. Hybrid mpi/openmp power-aware computing. In *24th IEEE International Parallel and Distributed Processing Symposium*, Apr. 2010.

[10] D. Li, D. S. Nikolopoulos, K. W. Cameron, B. R. de Supinski, and M. Schulz. Power-aware mpi task aggregation prediction for high-end computing systems. In *24th IEEE International Parallel and Distributed Processing Symposium*, Apr. 2010.

[11] J. Li and J. F. Martìnez. Dynamic power-performance adaptation of parallel computation on chip multiprocessors. In *12th International Symposium on High-Performance Computer Architecture*, Austin, Texas, Feb. 2006.

[12] OpenMP Architecture Review Board. OpenMP application program interface, May 2008. Version 3.0.

[13] S. E. Perl and W. E. Weihl. Performance assertion checking. *ACM SIGOPS Operating Systems Review*, 27(5):134–145, 1993.

[14] M. Rosing, R. Schnabel, and R. Weaver. The Dino parallel programming language. *Journal of Parallel and Distributed Computing*, 13(1):30–42, Sept. 1991.

[15] R. Springer, D. K. Lowenthal, B. Rountree, and V. W. Freeh. Minimizing execution time in MPI programs on an energy-constrained, power-scalable cluster. In *ACM Symposium on Principles and Practice of Parallel Programming*, Mar. 2006.

[16] J. S. Vetter and P. H. Worley. Asserting performance expectations. In *Supercomputing*, Nov. 2002.

[17] H. Zima, H. Bast, and M. Gerndt. SUPERB: A tool for semi-automatic MIMD/SIMD parallelization. *Parallel Computing*, 6(6):1–18, Jan. 1988.